

Evolutionary Algorithms for Learning of Mobile Robot Controllers

Frank Hoffmann, Oliver Malki, Gerd Pfister
Institute of Applied Physics
University of Kiel
Phone: +(49)-431-880-3934
Fax: +(49)-431-880-4608
E-Mail: hoefi@ang-physik.uni-kiel.de

Abstract- This paper presents an automatic design method for fuzzy systems using genetic algorithms. A flexible, compact coding scheme for the genetic representation of the fuzzy rule base is suggested. The method is applied to adapt the behaviour of a mobile robot implemented by means of a fuzzy logic controller. The mobile robot is tested on real world situations.

Keywords: Fuzzy control, genetic algorithms, mobile robot, autonomous agent

1 Introduction

The guiding principle of soft computing is the combination of neural networks, fuzzy logic (FL) and genetic algorithms (GAs) in hybrid systems in order to benefit from the advantages of each methodology. Thirty years ago Lotfi Zadeh [6] founded the theory of fuzzy sets, by extending the classical concept of a set. Unlike classical logic, in which elements either do or do not belong to a set, the degree of membership for elements of a fuzzy set can take on any value of the interval $[0, 1]$. Fuzzy logic offers a framework for representing imprecise, uncertain knowledge. Similar to the way in which human beings make their decisions fuzzy systems are using a mode of approximate reasoning, which allows them to deal with vagueness and incomplete information. Fuzzy control (FC) provides a flexible method to model the relationship among input information and control output. Fuzzy logic controllers (FLCs) prove their robustness with regard to noise and variations of system parameters.

Genetic algorithms (GAs) are optimisation methods inspired by the principles of natural evolution and genetics. GAs have been successfully applied to solve a variety of difficult theoretical and practical problems by imitating the underlying processes of evolution such as selection, recombination and mutation. Their capability of learning enables a GA to adapt a system to deal with any desired task, that can be described by a scalar objective function. In contrast to other specialised optimisation methods GAs work well across a broad spectrum of problems and require no problem specific knowledge.

Following the guideline of soft computing GAs have been widely used for automatic design of FLCs [2]. The basic idea is to learn the rules of a knowledge based system by artificial evolution. The design of a FLC by hand requires expert knowledge in order to formulate the fuzzy rules. This design process can be automated by the use of a GA, which adapts the rule base in order to achieve a desired control behaviour of the fuzzy system. In this contribution we present a messy coding scheme, which allows a compact and efficient genetic representation of the fuzzy rule base. The coding scheme enables fuzzy rules to be flexibly composed of a variable number of fuzzy terms.

It is known that GAs are generally suitable to adapt the behaviour of an autonomous agent. We apply our design method to evolve a FLC in order to implement the behaviour of a mobile robot. The controller learned in a simulation is tested in real world experiments in order to verify the approach.

2 Designing Fuzzy Logic Controllers using Genetic Algorithms

GAs are optimisation methods based on the model of natural evolution. A population of candidate solutions to the optimisation problem is evolved from one generation to the next in order to obtain slightly improved solutions according to an objective function. The quality function assigns each of the individuals a score, which is called *fitness*. Each of the candidate solutions is represented by a genetic string of parameters called *chromosome*. According to Darwin's principle the probability of reproduction increases for individuals with higher fitness. A new offspring is created by recombination and mutation of the chromosomes of the selected parents.

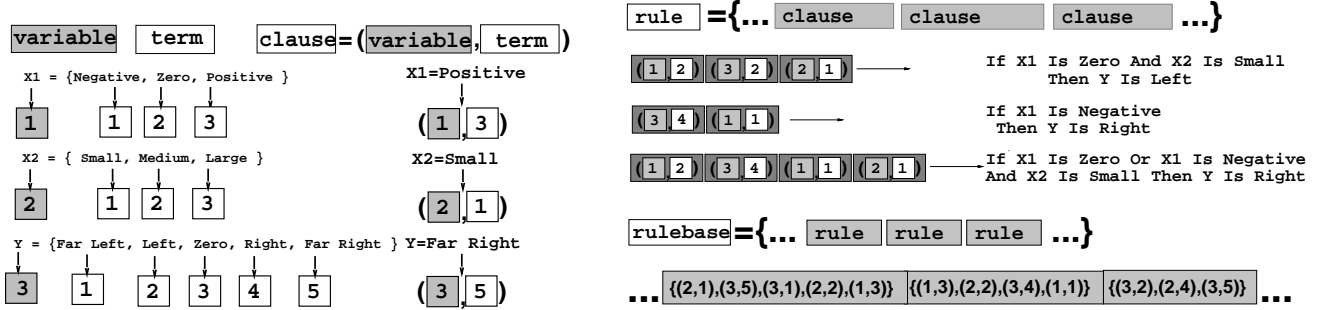


Figure 1: Messy coding of a fuzzy knowledge base

The design of a knowledge based fuzzy system requires the definition of fuzzy input and output variables. Linguistic concepts such as *Near*, *Far* given by a human expert are represented by fuzzy sets. The meanings of linguistic terms are specified by membership functions of the fuzzy sets. The rule base is composed of a set of fuzzy rules, which represent the expert's knowledge about the relationship among control input and output.

There are mainly two different ways to apply GAs to the design and the optimisation of FLCs. The first approach employs a GA to tune the membership functions of fuzzy sets according to a fitness function. This method requires a previously defined set of rules and is primarily useful to achieve an optimal performance of an already existing FLC. The parameters of linguistic terms such as centre and width of membership functions are encoded into chromosomes, which are processed by the GA.

In the second method the GA learns the fuzzy rule base. In this case a chromosome represents the label of linguistic terms in the consequence parts of fuzzy rules. For all conceivable combinations of input terms in the antecedent a control action has to be determined. For n input variables the rule base is represented by an n -dimensional rule matrix, in which each element contains the label of the corresponding output fuzzy term. There exist hybrid methods in which each rule possesses its own fuzzy sets, instead of using a common set of fuzzy terms for all rules. The use of coding schemes, in which rules and parameters of the underlying fuzzy sets are evolved simultaneously, enables the adaptation of the entire knowledge base.

Using a fixed coding scheme for the fuzzy rule base results in a significant drawback. The number of rules increases rapidly with a growing number of input variables. This negative effect reduces the computational efficiency and the robustness of the FLC. Therefore the optimisation problem for the GA becomes less and less feasible because of the high dimensional search space. More efficient and flexible genetic representations of the fuzzy rules and the rule base are necessary in order to evolve more complex FLCs by a GA. In imitation of messy genetic algorithms we suggest in the following a coding scheme for fuzzy rules of variable length and structural complexity.

A fuzzy rule is composed of several fuzzy clauses of the form:

$$\langle \text{variable} \rangle \text{ Is } \langle \text{fuzzy term} \rangle$$

in which the variable can either corresponds to an input or an output. Fuzzy clauses are the basic elements of our coding scheme. A fuzzy clause is encoded into a pair of integers, in which the first integer determines a variable and the second integer characterizes the label of the fuzzy term associated with that variable. The example depicted in Fig. 1 on the left has two input variables X_1 and X_2 each of them containing three fuzzy terms and one output variable Y with five fuzzy terms. A fuzzy clause like X_1 Is *Positive* is coded by a single gene in form of a pair $(1,3)$. The 1 determines X_1 as the fuzzy variable and 3 corresponds to the third fuzzy term *Positive* of X_1 . The pair $(3,5)$ for example represents the fuzzy clause Y Is *Far Right*.

Fuzzy sets are defined by membership functions given by a human expert and are not part of the optimisation process. In our approach the GA only learns the labels of linguistic terms but the membership functions of fuzzy sets are kept fixed. An extension of the coding scheme is possible, in which each clause defines its own membership functions, instead of using a common set of fuzzy terms for all clauses. This alternative representation encodes a fuzzy clause into a triple (n, x, σ) , in which n is the index of the variable, x the centre and σ the width of a triangular or gaussian membership function.

Any unordered set of fuzzy clauses with any number of elements forms a fuzzy rule. The right side of Fig. 1 depicts some possible examples of fuzzy rules, which contain different combinations of fuzzy clauses. In the first example three pairs of integers $(1,2)$ $(3,2)$ $(2,1)$ form the fuzzy rule:

If X_1 Is Zero **And X_2 Is Small **Then** Y Is Left**

Notice that the clauses of a fuzzy rule can be arranged in any order whatever. The combination **(3,2) (2,1) (1,2)** represents another possible coding of the same fuzzy rule. Reordering of genes in the chromosome can be advantageous, because it enables a messy GA to form tight and efficient building blocks according to the schema theorem.

Any set of clauses containing at least one input and one output variable can be mapped to a meaningful fuzzy rule. Input variables, for which no corresponding pair is present in the set of clauses, are simply omitted in the antecedent of the rule. The set **(3,4) (1,1)** is underspecified, because it does not contain a pair with the integer **2** in the first place. Therefore the variable X_2 does not occur in the rule.

If X_1 Is Negative Then Y Is Right

Multiple clauses for the same variable are combined with a fuzzy or. The set **(1,2) (3,4) (1,1) (2,1)** is overspecified because it includes two clauses for variable X_1 , which results in the fuzzy rule.

If X_1 Is Zero Or X_1 Is Negative And X_2 Is Small Then Y Is Right

Finally the rule base is composed of a variable number of rules arranged in any order whatever as depicted in the bottom of Fig. 1 on the right. Rules cover large regions of input space, if they only contain a subset of all input fuzzy variables. The size and the structural complexity of the rule base can therefore be reduced. The flexible coding scheme allows a compact representation in order to model the relationship among control input and output by fuzzy rules.

Messy GAs employ a modified crossover operator for recombination of genetic material. The positions of cuts on the two parent chromosomes can be chosen regardless of each other, because the meaning of a gene is independent of its location in the genetic string. For the same reason the resulting pieces can be spliced in any order. In our messy coding scheme cut and splice operations act on two levels. On the lower level cut and splice operation combines fuzzy clauses to form new fuzzy rules. A matching operation among parent chromosomes is applied in order to restrict recombination to those fuzzy rules that become active for similar input situations.

On the upper level cut and splice operators merge sets of fuzzy rules to create new rule bases. Two parents that have already learned good control rules for different regions of input space, benefit from each other if they exchange parts of their rulebases. This complies with the general idea of crossover to combine genetic material of solutions from promising regions of search space. See [3] for more details on the coding scheme and genetic operators.

3 Adapting the Behaviour of an Autonomous Agent

Among many other applications fuzzy rule based systems have been used for the control of mobile robots [4][5]. We applied our method to adapt a reactive behaviour of an autonomous agent, which is implemented by means of a FLC. A mobile robot, equipped with five ultrasonic sensors to perceive its environment, is given the task to reach a specified goal point while avoiding obstacles on its way.

The majority of examples to the design of FLCs using GAs applied the method to standard control problems, in which the performance of the FLC is evaluated in a simulation of the process. For a number of reasons GAs are not very suitable for on-line learning of controllers using the real world process. First of all a GA works with a population of controllers, which requires a large number of fitness evaluations and is therefore very time consuming. Secondly all of the FLCs have to be tested under the same initial conditions for a representative variety of process states. In our application a human operator has to intervene after the evaluation of a controller in order to bring the mobile robot back to the starting point.

It is therefore necessary to evaluate the fitness of individuals in a computer simulation of the dynamic process. FLCs are tested in virtual environments using a simplified model of the robot's sensors based on idealistic assumptions with respect to sonar distance measurements. Using a simulation to adapt the control behaviour is only justified if afterwards the FLC proves enough robustness when confronted with reality.

Each controller C_k is tested on the simulated robot for a set of thirty virtual environments E_i , which differ in the positions of start, goal point and obstacles. A simulated run of the robot stops either if a collision occurs, a maximum number N_{\max} of control steps is exceeded or the goal point is reached. For each case the controller receives a different amount of reinforcement.

$$\begin{aligned}
 R_c(C_k, E_i) &= N_c / N_{\max} && \text{collision} \\
 R_d(C_k, E_i) &= 1 - d_{\min} / d_0 && \text{no collision} \\
 R_t(C_k, E_i) &= 1 && \text{goal point reached}
 \end{aligned} \tag{1}$$

In case of a collision the controller receives only a small reward R_c proportional to the number of steps N_c made so far. If a collision is avoided but the goal point is missed the controller is given an additional reward R_d , depending on how close the robot approached the goal point. In Eq. 1 d_0 is the distance between the start and the goal point while d_{\min} is the minimum distance between robot and the goal point, which occurred during N_{\max} steps. Notice that the controller receives the sum $R_c + R_d$ of both rewards. If the goal point is reached the controller receives the maximum reward $R_c + R_d + R_t$.

If the GA uses a static fitness function the population converges prematurely to FLCs only adapted to the primary objective of obstacle avoidance. In order to evolve controllers able to cope with both tasks simultaneously a dynamically weighted fitness function is suggested. Each objective contributes a fixed amount of fitness, which is shared among all members of the population. The fitness $F(C_k)$ of an individual C_k is composed of the three different sources of reinforcements from Eq. 1 each one previously divided by the sum of scores all the other individuals obtained on the same objective.

$$F(C_k) = \sum_{i=1} \left(R_c(C_k, E_i) / \sum_k R_c(C_k, E_i) + R_d(C_k, E_i) / \sum_k R_d(C_k, E_i) + R_t(C_k, E_i) / \sum_k R_t(C_k, E_i) \right) \quad (2)$$

In the first generations only few individuals manage to reach the goal point in about one or two environments. These controllers achieve relatively high fitness values, because they have to share their pay-offs R_t with only few competitors. They obtain a good chance for selection, even if they fail in obstacle avoidance more often than an average individual. During the course of evolution more and more controllers appear that are adapted to both objectives.

In order to achieve a robust control behaviour the FLCs have to be evaluated on a sufficient number of representative environments. When using a fixed set of environments for fitness evaluation the controllers showed good performance on the test cases presented during evolution but often failed in previously unseen configurations of obstacles. In imitation of the relationship among predator and prey populations in natural evolution we take a second GA that evolves a population of test environments simultaneously with the FLCs. Parameters of the environments such as positions of obstacles, start and goal point are encoded in a chromosome using a binary representation. Environments achieve high fitness values when lot of controllers failed in them. The fitness $F(E_i)$ of an environment E_i is the inverse of the sum of rewards $R(C_k, E_i)$, which controllers C_k achieved in it.

$$F(E_i) = \frac{1}{\sum_k R_c(C_k, E_i) + R_d(C_k, E_i) + R_t(C_k, E_i)} \quad (3)$$

By using coevolution of test sets FLCs are more often evaluated in environments that turned out to be difficult for the robot in the past. A more robust control behaviour emerges because FLCs can no longer benefit from easy environments, which die out in the course of evolution.

4 Experimental Results with the Mobile Robot

The FLC, which is presented in the following, obtains as input the distance measurements from the five ultrasonic sensors, the distance and the direction to the goal point. The inference mechanism maps these informations to the steering radius of the robot, which plays the role of control output. The fuzzy sets of all linguistic variables are given in advance and are not part of the optimisation. The FLC evolved by the GA is composed of 71 rules, which is only a small fraction of the 46875 rules required when the rule base is encoded using a conventional coding scheme.

The decision on the control action is only based on the current input and is made irrespective of previous sensor information. Nevertheless the controller has learned to turn the robot in a corridor if the goal point is located in opposition to the robot's heading. The agent is able to stop the manoeuvre in time if the corridor is too narrow for a turn. The robot moves on keeping its original direction and turns around when there is space enough.

The FLC is tested on the mobile robot in real-world experiments under two different aspects. First of all it is important to verify that the behaviour of the real robot is comparable with the one demonstrated in the simulation. Although distance measurements based on sonar signals are very unreliable and despite many idealistic assumptions in our model there turned out to be only slight differences in control behaviour among simulation and reality.

The second matter of interest in the experiments was the agent's capability to deal with previously unseen situations, especially when there is a conflict between the objectives of obstacle avoidance and reaching the goal

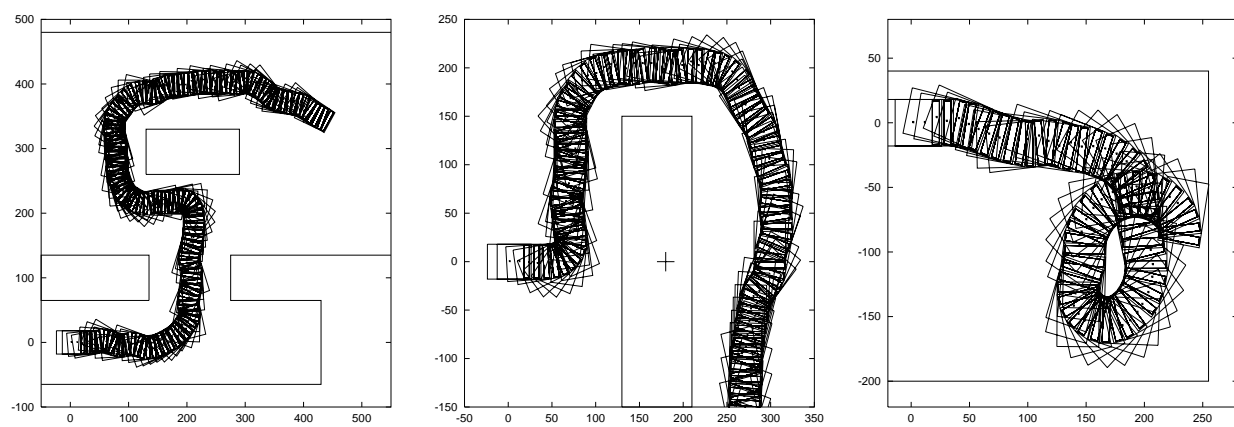


Figure 2: Experiments with the mobile robot

point. Fig. 2 shows on the left side an experiment with the real world robot in a theatre at our institute. The environment differs from those presented during evolution in such a way that the corridor branches with the exit located at the left side. The test environments contained neither branches nor dead-ends and the corridor's exit was always in a straight line with the robots heading. Nevertheless the agent is successful in passing through the gap on the left side and in reaching the goal point.

In all of the test environments the goal point was always reachable for the robot. In the middle of Fig. 2 an experiment is depicted, in which the goal point is located inside an obstacle. The robot drives around the obstacle and keeps close to the wall, which shows that the autonomous agent is able to take the higher priority of collision avoidance into account. Finally we placed the agent in a corridor, which ends up with a barrier right in front of the goal point; a situation completely different from test environments. The experiment depicted in Fig. 2 on the right shows that the robot collides with the wall, although there is enough space for an evasive action. This example demonstrates that a control behaviour adapted in selected environments can not be transferred to every conceivable situation.

5 Conclusions

We presented an automatic design method for FLCs based on GAs. The messy coding scheme uses a compact and flexible genetic representation of the fuzzy rule base, which reduces the size and complexity of the optimisation task for the GA. We applied our method to evolve a FLC for the control of a mobile robot. Although the control behaviour is adapted in virtual environments the FLC proves enough robustness to compensate differences between simulation and reality. The robot is successful in environments that are similar to those presented during evolution, while it fails in situations that differ substantially.

References

- [1] Andrea Bonarini, "Evolutionary Learning of Fuzzy Rules: Competition and Cooperation", *Fuzzy Modelling: Paradigms and Practice*, W. Pedrycz (Ed.), Kluwer Academic Press, Norwell, MA, 1996
- [2] Oscar Cordon, Francisco Herrera, Manuel Lozano, "A Classified Review on the Combination Fuzzy Logic Genetic Algorithms Bibliography", Tech. Report 95129, <http://decsai.ugr.es/herrera/fl-ga.html>, 1995
- [3] Frank Hoffmann, Gerd Pfister, "Learning of a Fuzzy Control Rule Base Using Messy Genetic Algorithms", *Genetic Algorithms and Soft Computing*, "Studies in Fuzziness", Physica-Verlag, Heidelberg, 1996
- [4] Alessandro Saffiotto, Enrique H. Ruspini, Kurt Konolige, "A Fuzzy Controller for Flakey an Autonomous Mobile Robot", *Proc. 3. Dortmund Fuzzy-Tage*, Springer-Verlag, pp. 3-12, 1993
- [5] Hartmut Surmann, Jörg Huser, Liliane Peters, "Fuzzy System for Indoor Mobile Robot Navigation", *Proc. of the Fourth Int. Conf. on Fuzzy Systems*, Yokohama, pp. 83-86, 1995
- [6] Lotfi A. Zadeh, "Fuzzy Sets", *Journal of Information and Control*, vol. 8, pp. 338-353, 1965